SHA-3 Instruction Set Extension for A 32-bit RISC Processor Architecture

Ahmed S. Eissa, Mahmoud A. Elmohr, Mostafa A. Saleh,
Khaled E. Ahmed, Mohammed M. Farag
Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt
eissa.s.ahmed@gmail.com, mahmoud.a.elmohr@ieee.org, moustafa.i.saleh@gmail.com, k.e.elsayed@ieee.org, mmorsy@alexu.edu.eg

***Abstract*—The Secure Hash Algorithm (SHA-3) is a cryptographic hash function widely used in most security applications. The execution of the SHA-3 function is computationally intensive on lightweight embedded RISC processors. In this work, we advance a SHA-3 Instruction Set Extension (ISE) to improve its performance on a 32-bit MIPS processor. Two ISE approaches are proposed, namely native datapath and coprocessor-based ISEs. The ISE is developed with the aid of Codasip Studio, and the extended processor is implemented and benchmarked on a Xilinx Virtex-6-XC6VLX75t FPGA. The benchmarking results exhibit a 21% and 43% increase in the execution speed of the SHA-3 algorithm on the MIPS processor at the expense of 9% and 26% resource overheads for the native datapath and coprocessor-based ISEs, respectively.**

***Index Terms*—SHA-3, Instruction Set Extension, Application-Specific Instruction Set Processor, MIPS, RISC.**

1. Introduction

In 2015, the National Institute of Standards and Technology (NIST) announced Keccak as the new Secure Hash Algorithm (SHA-3) standard [1]. Keccak is a sponge construction with an arbitrary input and output length that repeatedly performs five permutations on a state array of 5×5 lanes each 64-bit length. Each permutation consists of the iteration of a simple round function limited to bitwise XOR, AND and NOT and rotations [2]. The execution of the SHA-3 function is computationally intensive on lightweight embedded RISC processors because of the large size of the SHA-3 state. Instruction Set Extensions (ISEs) can be applied to improve performance of the SHA-3 computation on such processors. In [3], various ISEs have been presented for a 16-bit PIC24 microprocessor to accelerate the computation of the five SHA-3 candidates. In this work, we advance two ISEs to speedup the SHA-3 computation on a lightweight 32-bit MIPS processor [4] with the aid of Codasip Studio, an automated ASIP development environment [5].

2. SHA-3 Instruction Set Extensions

A custom 32-bit five-stage pipelined MIPS processor is implemented that only includes logic and arithmetic instructions without overflow, memory instructions load and store word, branch and jump instructions. Two ISE variants are advanced: the native datapath and the coprocessor-based ISEs. The syntax of the added instructions and their high-level description are provided in Table 1 and Table 2.

Native Datapath SHA-3 ISE: In this approach, the SHA-3 bottlenecks are tackled by introducing minor microarchitectural modifications to the MIPS processor native datapath. Four new instructions are added to the MIPS instruction set architecture (ISA): `AndNot`, `Rot1`, `Rot2` and `MFLeast`. To enable the ISE, the MIPS execute stage is modified as shown in Figure 1. For the rotation instructions, two special internal registers are inserted to hold the intermediate value for the 64-bit rotation. A multiplexer is inserted to switch between the most and least significant bits in different cycles. For the AndNot instruction, an inverter is inserted as additional input to the ALU source multiplexer.

Coprocessor-based SHA-3 ISE: In this approach, the SHA-3 bottlenecks are resolved by adding a coprocessor that operates on multiple inputs at once. Five new instructions are added: `LWAu`, `XOR5`, `Chi`, `Rot`, and `SWLeast`. The microarchitecture of the SHA-3 coprocessor is shown in Figure 2. To keep the MIPS register file structure unchanged, five auxiliary registers are added to the decode stage to supply parallel inputs to the Co-ALU. Since only 32-bit data can be stored in the memory in one cycle, the extra bits are held in an internal register to be stored in memory in another cycle. Auxiliary register address is assigned by the instruction bits[18:16] which are also used as an immediate index for the `Chi` instruction.

3. Results and Evaluation

The extended MIPS processor is synthesized and tested on a Xilinx Virtex-6 XC6VLX75t FPGA. To illustrate the enhancements gained by the proposed SHA-3 ISEs, our extended architectures are compared to the basic MIPS processor in terms of the SHA-3 computation time measured in the number of cycles to hash one byte, processor area in the number of FPGA slices, and memory utilization in bytes. Table 3 provides a comparison between our basic MIPS, the native datapath and coprocessor-based extended MIPS, and the Keccak-extended PIC24 processor presented in [3].

The results show a reduction of the SHA-3 program execution time by 18% and 30% which is equivalent to a

TABLE 1. NATIVE DATAPATH ISE INSTRUCTIONS

Instruction	Description
AndNot	AndNot $destination, $source1, $source2 $destination ← (∼ $source1)&$source2
Rot1	Rot1$source1,$source2 **if** $source2 < 32 **then** Most ← $source1 ≪ $source2 Least ← $source1 ≫ (32 − $source2) **else** Least ← $source1 ≪ ($source2 − 32) Most ← $source1 ≫ (64 − $source2) **end if**
Rot2	Rot2 $destination, $source1, $source2 **if** $source2 < 32 **then** Least ← Least\|($source1 ≪ $source2) Most ← Most\|($source1 ≫ (32 − $source2)) **else** Most ← Most\|($source1 ≪ ($source2 − 32)) Least ← Least\|($source1 ≫ (64 − $source2)) **end if** $destination ← Most
MFLeast	MFLeast $destination $destination ← Least

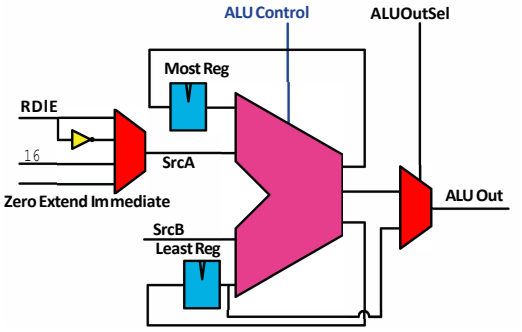

Figure 1. MIPS ALU modifications to enable native datapath ISE

TABLE 2. SHA-3 COPROCESSOR-BASED ISE INSTRUCTIONS

Instruction	Description
LWAu	LWAu $destinationAu, #offset($base) Executed by the regular ALU: $destinationAu ← memory[$base + #offset]
XOR5	XOR5 #offset($base) Executed by the additional ALU: Result ← $Au0 ⊕ $Au1 ⊕ $Au2 ⊕ $Au3 ⊕ $Au4 Executed by the regular ALU: memory[$base + #offset] ← Result
Chi	Chi #index, #offset($base) Executed by the additional ALU: Result ← $Au[#index] ⊕ ((∼ $Au[#index + 1])&Au[#index + 2]) Executed by the regular ALU: memory[$base + #offset] ← Result
Rot	Rot #offset($base) Executed by the additional ALU: {Most, Least} = {$Au0, $Au1} ≪ $Au2 Executed by the regular ALU: memory[$base + #offset] ← Most
SWLeast	SWLeast #offset($base) Executed by the regular ALU: memory[$base + #offset] ← Least

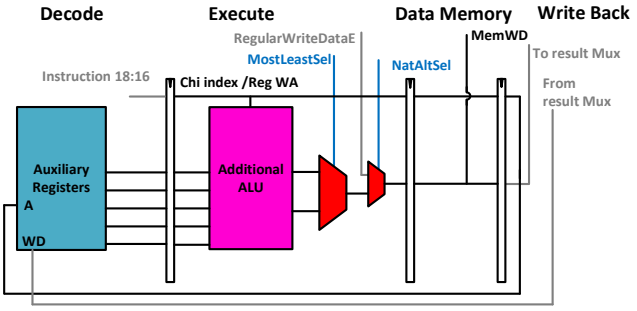

Figure 2. SHA-3 coprocessor datapath

21% and 43% speedup in the computation; and a reduction in the code size by 4% and 6% for the native datapath and coprocessor-based ISEs, respectively. Both extended MIPS processors exhibit relatively large area overheads of 9% and 26% because only a small subset of the reference MIPS ISA is implemented. Comparing our results to the PIC24 ISE results depicts an equivalent performance improvement, as a percentage, and a significant reduction in the total code size; whereas the PIC24 ISE achieves a better code size reduction percentage and a smaller absolute execution time.

TABLE 3. IMPLEMENTATION RESULTS

	Execution time Cycles/byte	Area # of Slices	Code memory Bytes
Reference MIPS	254	6074	1928
Native ISE	209 (82%)	6595 (109%)	1852 (96%)
Coprocessor ISE	178(70%)	7645 (126%)	1812 (94%)
PIC24 [3]	188	–	3480
PIC24+ISE [3]	132 (70%)	–	2415 (69%)

4. Conclusions

In this paper, we presented two ISEs, namely native datapath ISE and coprocessor-based ISE, to improve performance of the SHA-3 computation on a lightweight 32-bit MIPS processor. The extended MIPS processor is synthesized on a Virtex 6 FPGA, the native datapath and coprocessor-based ISEs speedup the execution of the SHA-3 algorithm by 21% and 43%, respectively. The resource utilization of the extended MIPS processor, however, is increased by 9% and 26% for the proposed ISEs. Despite that the proposed SHA-3 ISEs are specifically developed for the MIPS ISA, they can be generally applied to other 32-bit RISC processor architectures. As a future work, we plan to investigate further performance improvements of the SHA-3 standard on other embedded processor architectures.

References

[1] Penny Pritzker and Patrick D Gallagher. SHA-3 standard: Permutation-based hash and extendable-output functions'. *Information Tech Laboratory National Institute of Standards and Technology*, pages 1–35, 2014.

[2] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3:30, 2009.

[3] Jeremy Hugues-Felix Constantin, Andreas Peter Burg, and Frank K Gurkaynak. Investigating the potential of custom instruction set extensions for SHA-3 candidates on a 16-bit microcontroller architecture. Technical report, Cryptology ePrint Archive, 2012.

[4] David Harris and Sarah Harris. *Digital Design and Computer Architecture*. Elsevier, 2012.

[5] Codasip Ltd. www.codasip.com.